

DbproExeFormat

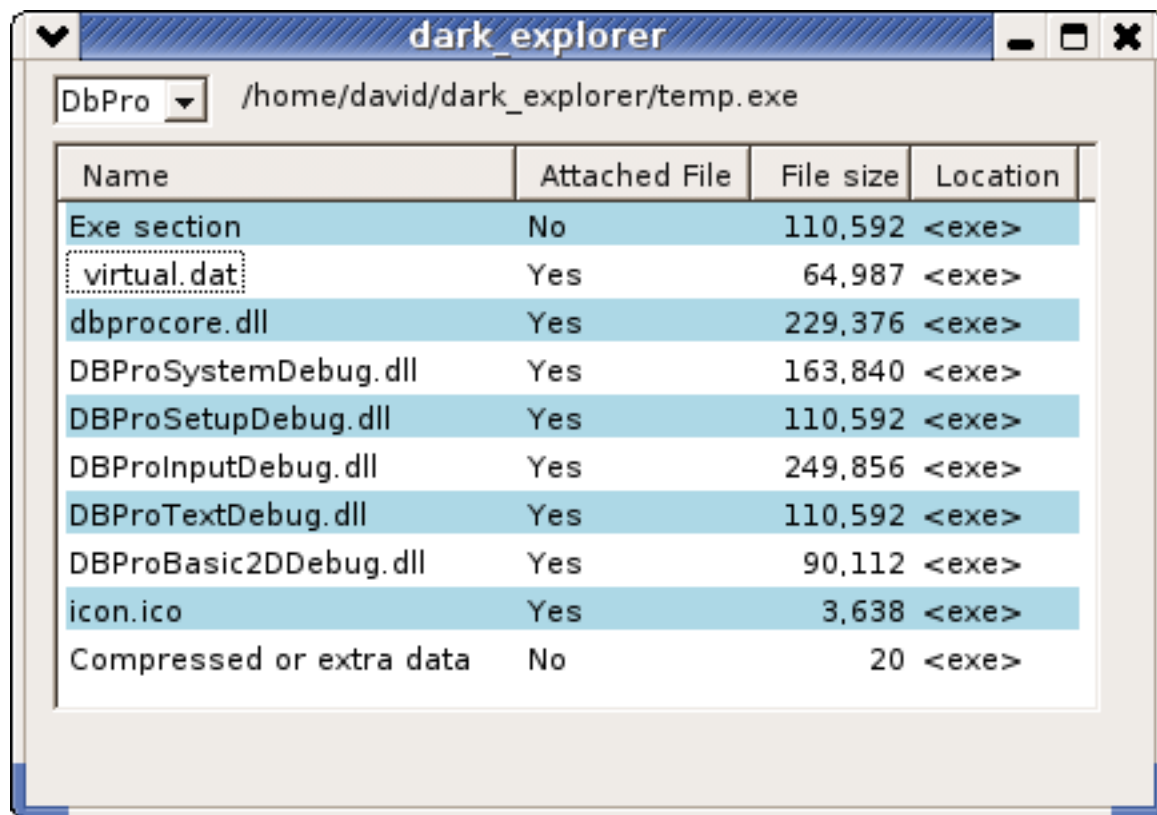
Darkbasic pro and classic exe format information. Information based on exes produced by dbpro 5.9 and dbc 1.13

Darkbasic exes consist of a standard win32 exe with files and data appended onto the end.

When the dbpro exe is run it creates a dbpdata directory in the windows temp dir. If there is an existing dbpdata dir a number is appended to the end and incremented, dbpdata1, dbpdata2 etc. until a name that doesn't exist is found. The exe then extracts the appended files into this dbpdata dir. The extracted files consist of dbpro dlls, plugin dlls, attached media and the _virtual.dat file. The _virtual.dat file contains among other things the compiled dbpro code.

Dbc exes do not extract any appended files or data when run. Media is loaded without first extracting it.

The following image shows the contents of an example dbpro exe.



The screenshot shows a window titled "dark_explorer" with a menu bar containing "DbPro" and a file path "/home/david/dark_explorer/temp.exe". Below the menu bar is a table with four columns: "Name", "Attached File", "File size", and "Location". The table lists the following items:

Name	Attached File	File size	Location
Exe section	No	110,592	<exe>
virtual.dat	Yes	64,987	<exe>
dbprocore.dll	Yes	229,376	<exe>
DBProSystemDebug.dll	Yes	163,840	<exe>
DBProSetupDebug.dll	Yes	110,592	<exe>
DBProInputDebug.dll	Yes	249,856	<exe>
DBProTextDebug.dll	Yes	110,592	<exe>
DBProBasic2DDebug.dll	Yes	90,112	<exe>
icon.ico	Yes	3,638	<exe>
Compressed or extra data	No	20	<exe>

Standard win32 exe

The start of a darkbasic exe is a standard win32 exe. The exe section of all dbpro exes compiled with the same dbpro version appear to be very similar for all compiled programs. The only time this section appears to differ is when you modify the exe icon or version information. The most reliable method to find the size of this section is to use some knowledge of the windows exe format. Wotsit.org has several documents that describe the windows exe format.

Here is a simple function that returns the size in bytes of an exe section of a db exe.

```
function get_exe_size(fileName as string)
    `Returns the size in bytes of an exe section of a db exe
    file_num = 1
    open to read file_num, fileName
        skip bytes file_num, 60
        read long file_num, e_lfanew
    close file file_num
    open to read file_num, fileName
        skip bytes file_num, e_lfanew + 6
        read word file_num, numberOfSections
        skip bytes file_num, 240
        for i = 1 to numberOfSections
            skip bytes file_num, 16
            read long file_num, size
            read long file_num, pos
            skip bytes file_num, 16
        next i
    close file file_num
    inc size, pos
endfunction size
```

Db does not use such a method. Instead it writes the length of the exe section as a 4 byte int at the very end of the exe. This method can cause problems when you change the size of the exe, for instance when you change the exe icon or version information.

Attached files

Now that we know the size of the exe section we can skip it to get straight to the appended data. The format used to append the files is very simple and easy to read. Files are stored with a filename first then the filedata. Both the filename and filedata are preceded with a 4byte int which tells you how long the following filename or filedata is. With dbc exes filenames are terminated with a null byte but dbpro does not.

- int namelength
- name
- int datalength
- data

The following image shows an example .pck file loaded in a hexeditor.

0000000000	08 00 00 00	74 65 73 74 2e 70 6e 67	5a 00 00 00	test.pngZ...	
0000000016	89 50 4e 47 0d 0a 1a 0a 00 00 00 0d	49 48 44 52		.PNG.....IHDR	Name length
0000000032	00 00 00 04 00 00 00 04 08 02 00 00	00 26 93 09		&..	Filename
0000000048	29 00 00 00 21 49 44 41 54 08 d7 4d c9 b1 0d 00			)...!IDAT.�M��..	Data length
0000000064	30 08 c0 b0 d0 ff 7f 36 4b 07 36 4b ae 02 ba c2			0.����.6K.6K�.��	File data
0000000080	ab 66 e6 cf cd 05 95 2e 17 ee 0e 0c 26 81 00 00			�f����....�..&...	
0000000096	00 00 49 45 4e 44 ae 42 60 82	0b 00 00 00	61 6e	..IEND�B`.....an	
0000000112	6f 74 68 65 72 2e 70 6e 67	56 00 00 00	89 50 4e	other.pngV....PN	
0000000128	47 0d 0a 1a 0a 00 00 00 0d	49 48 44 52 00 00 00		G.....IHDR...	
0000000144	04 00 00 00 04 08 02 00 00 00 26 93 09 29 00 00			&..)	
0000000160	00 1d 49 44 41 54 08 d7 63 60 60 60 f8 ff ff ff			..IDAT.�c``�����	
0000000176	ff ff ff 19 18 18 18 21 14 04 30 31 e0 02 00 7f			����....!...01�...	
0000000192	1e 08 fb fc cc 15 6c 00 00 00 00 49 45 4e 44 ae			..����.l....IEND�	
0000000208	42 60 82			B`.	

Continue reading files while namelength is > 0 and < 255 or you reach the end of the file.

Some code that lists the files in an exe, requires the get_exe_size function above.

```

sync on : sync
exeFile as string
exeFile = "temp.exe"
exeFileSize = file size(exeFile)
exeSize = get_exe_size(exeFile)
text 0, y, "exe section"
text 250, y, str$(exesize)
inc y,15
open to read 1, exeFile
  skip bytes 1, exeSize : `skip exe section
  namesize = 1
  repeat
    name$ = ""
    read long 1, namesize
    if namesize > 0 and namesize < 255
      for i = 1 to namesize
        read byte 1, b
        name$ = name$ + chr$(b)
      next i
      read long 1, datasize
      skip bytes 1, datasize
      text 0, y, name$
      text 250, y, str$(datasize)
      inc y, 15
    endif
  until namesize = 0 or namesize > 255
  `check for extra data
  if file end(1) = 0
    datasize = 4 : `namesize was read earlier so start at 4 so those 4 bytes
are counted
    repeat
      read byte 1, b
      if file end(1) = 0 then inc datasize
    until file end(1)
    text 0, y, "extra data"
    text 250, y, str$(datasize)
  endif
close file 1
sync

```

wait key

With dbpro exes after the files is 20 bytes of extra data. The last 4 bytes of this extra data is the size of the exe section stored in a 4 byte int. With dbc the extra data is only 4 bytes and contains just the size of the exe section.

Dbpro specific information

The above code will only let you view the contents of a normal dbpro exe, through the options in the IDE you can instruct the compiler to compile one of several different types of dbpro exe. For all of these types the standard exe section is identical. All types can also contain attached media files. Any media files will have a "media\" prefix on the filename. The dbpro media loading commands automatically check for attached media so you can add and remove media from the exe without problems. It is possible to change a compiled exe of one type into any of the other types.

- Normal Executable - The default option and by far the most popular, this is the format discussed above.
- Executable with .Pck file - Instead of being appended to the exe the files are placed into a .pck file.

The above types can also be optionally compressed.

Normal Executable

Almost all dbpro exes you find will be in this format as it's the default option and the most practical. Consists of a standard exe section followed by the attached files and ends with the extra data.

Executable with .Pck file

Instead of appending the files to the exe they are stored in a .pck file. The .pck file will have the same name as the program just with a .pck extension. Files are stored in the .pck file in the same format as they are appended to the exe. Pck files do not have the extra 20 bytes of data at the end. To identify this type of exe check for a .pck file or see if the exe section is the same size as the exe.

Compressed data

The attached files are compressed to reduce the size of the exe. Both exes and .pck files can be compressed. With compressed data there is first an uncompressed file, compress.dll. This dll contains functions required to decompress the data. Following compress.dll is the compressed data. With an exe this is followed by the 20 bytes of extra data. The two functions exported in compress.dll look like this,

```
HGLOBAL decompress_block(void* data, int datalength)
HGLOBAL compress_block(void* data, int datalength)
```

Using these two functions is fairly simple but requires the following winapi functions.

- [GlobalAlloc](#)
- [GlobalLock](#)

- [GlobalSize](#)

To decompress a compressed exe,

- 1. Load the exe as normal, and extract compress.dll
- 2. Following compress.dll is the compressed data, load it to memory ignoring the last 20bytes of extra data.
- 3. Call decompress_block passing it a pointer to the compressed data and its size
- 4. decompress_block returns a handle to a memory object, use GlobalLock on this handle to get normal memory pointer.
- 5. Use GlobalSize on the handle to find the size of the decompressed data.
- 6. Write a normal dbpro exe section to disk, then the uncompressed data followed by the 20 bytes of extra data.

To compress an uncompressed exe,

- 1. Source compress.dll from the dbpro compiler\plugins directory or another source.
- 2. Load a dbpro exe into memory discarding the exe section and 20 bytes of extra data.
- 3. Call compress_block passing it a pointer to the uncompressed data and its size.
- 4. compress_block returns a handle to a memory object, use GlobalLock on this handle to get normal memory pointer.
- 5. Use GlobalSize on the handle to find the size of the compressed data.
- 6. Write a normal dbpro exe section to disk, followed by the compress.dll you used as a normal attached file. Then write the compressed data followed by the extra data.

Dealing with .pck files is similar just with no exe section or extra data.

compress.dll code injection trick

We can inject our own compress.dll as the first attached file in a normal dbpro exe or .pck file. Then when the dbpro exe is run it will call our decompress_block function passing it the already decompressed data. A simple C example that shows a message box asking the user if they want to continue running the exe.

```
DLLIMPORT HGLOBAL decompress_block(void* data, DWORD dataSize)
{
    HGLOBAL block;
    if (IDYES == MessageBox(GetActiveWindow(), "Continue running exe?", "",
    MB_YESNO))
    {
        //for the dbpro exe to run it needs to receive the data as if it has been
        decompressed
        //by the real compress.dll. Since the dbpro exe is not compressed we just
        need to allocate
        //some memory, copy the already uncompressed data to it then return the
        HGLOBAL to dbpro
        block = GlobalAlloc(GMEM_FIXED, dataSize);
        if (block)
        {
            CopyMemory(block, data, dataSize);
        }
    }
}
```

```

    }
    else
    {
        //Returning 0 to dbpro will tell it an error has occurred and the dbpro
exe will silently exit
        block = (HGLOBAL) 0;
    }
    return block;
}

```

This allows us to do some interesting things including,

- Show a user friendly message box if the correct version of DirectX isn't installed.
- Show a display mode dialog and alter the `_virtual.dat` file in memory so the exe starts in the correct mode.
- Convert dbpro to a runtime style system with small exes by sourcing the dlls from the dbpro install dir.

[dark injector](#) is a simple program to inject a custom compress.dll into a dbpro exe.

[exe shrink](#) is an implementation of a runtime style system that uses a custom compress.dll to source the dlls the exe needs from an external location that can be shared by many exes.

_virtual.dat

Other than the standard exe section which changes when you use a different icon or version information `_virtual.dat` is the only file that differs between dbpro exes. You can therefore conclude the compiled code and other settings are stored in the `_virtual.dat` file.

At the very start of the file are 4 4byte integers. These four ints contain the settings for the initial display mode in this order.

- windowMode
- displayWidth
- displayHeight
- displayDepth

The values of the width, height and depth are obvious, the displayMode values are as follows.

- 0 = hidden
- 1 = windowed
- 2 = windowed desktop
- 3 = full exclusive
- 4 = windowed fullscreen

By modifying these 4 ints it's possible to change the initial display mode of a compiled exe.

Unfortunately lots of people for some reason ignore the display mode settings in the IDE and use SET DISPLAY MODE in their code instead. Setting the initial display mode will then not change the display mode, however you can set the initial display mode to the same as the mode used by SET DISPLAY MODE, sometimes this can reduce monitor flickering when the exe starts.

After the display settings is a null terminated string that is used as the window caption. Changing this string is easy if the replacement text is the same length, just swap the characters for the new ones. It is also easy to replace with a shorter string, just swap the characters and add a null character at the end. Leave the extra characters from the original string in place so the _virtual.dat file remains the same length. Sometimes it is even possible to replace it with a longer string, just overwrite the data following the original string.